САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра математической теории экономических решений

Свиридов Глеб Николаевич

Выпускная квалификационная работа бакалавра

Корректировка предварительного плана при изменении случайных параметров условий и целевой функции в задачах стохастического программирования.

Направление 010400

Прикладная математика и информатика

Научный руководитель,

старший преподаватель,

Тумка О.А.

Санкт-Петербург

2017

Содержание

Введение.

В настоящее время задача максимизировать прибыль на производстве очень популярна. Эта проблема достаточно легко разрешима для задач линейного программирования, если случай, где цена, стоимость товаров и другие факторы фиксированы и не меняются (детерминированный случай) **[1]**. Более трудной и актуальной является задача, когда стоимостные параметры в системе линейных уравнений и в целевой функции могут отклоняться, меняться на другую величину со случайным распределением значений **[2, 3]**. В рамках этой работы рассматривается задача, когда параметры будут меняться на случайные с учётом изменения изначального плана, т.е. требуется корректировка дальнейшего плана производства. Таким образом, оценка будущего состояния, когда появляются случайные отклонения в производственном процессе, также является составной частью планирования, что выявляет возможные нарушения предварительного планирования и вносит корректировку в цепочку принятия решений для поставленной производственной задачи.

Глава 1. Постановка задачи.

Обычно задача производства представляется в виде задачи линейного программирования (далее для краткости – ЗЛП).

ЗЛП состоит в том, что необходимо найти наибольшее(наименьшее) значение функции. Данная функция называется целевой, и она ограничивается некоторой областью значений.

Формулируется задача линейного программирования в общем виде следующим образом:

- Имеется целевая (линейная) функция:

$$z(x) = \sum_{k=1}^{m} c_k x_k \tag{1.1}$$

- Задана система линейных равенств (неравенств):

$$\sum_{k=1}^{m} a_{ik} x_k = b_i$$
$$x_k > 0 \tag{1.2}$$
$$i = 1 \ldots j$$

$$\sum_{k=1}^{m} a_{ik} x_k \leq b_i$$
$$x_k > 0 \tag{1.3}$$
$$i = j+1 \ldots n.$$

Мы будем рассматривать сначала детерминированный случай ЗЛП.

После этого проводим серию из *n* испытаний, в каждом из которых каждый раз будут меняться коэффициенты a_{ik}, b_i, c_k в некотором диапазоне изменений коэффициентов.

Перепишем ЗЛП с изменёнными параметрами:

Пусть мы решили детерминированный случай и получили $x_1 ... x_n$, тогда коэффициенты изменяются следующим образом:

- Целевая функция:

$$z(x;\omega) = \sum_{k=1}^{m} c_k(\omega) x_k \qquad \textbf{(1.4)}$$

- Система линейных равенств(неравенств):

$$\begin{aligned} &\sum_{k=1}^{m} a_{ik}(\omega) x_k = b_i(\omega) \\ &x_k > 0 \\ &i = 1...j \end{aligned} \qquad \textbf{(1.5)}$$

$$\begin{aligned} &\sum_{k=1}^{m} a_{ik}(\omega) x_k \le b_i(\omega) \\ &x_k > 0 \\ &i = j{+}1...n. \end{aligned} \qquad \textbf{(1.6)}$$

где

$$\begin{aligned} \boldsymbol{a_{ik}(\omega)} &= \boldsymbol{a_{ik}} \pm \Delta \boldsymbol{a_{ik}(\omega_i)} \\ \boldsymbol{c_k(\omega)} &= \boldsymbol{c_k} \pm \Delta \boldsymbol{c_k(\omega_i)} \\ \boldsymbol{b_i(\omega)} &= \boldsymbol{b_i} \pm \Delta \boldsymbol{b_i(\omega_i)} \end{aligned} \qquad \textbf{(1.7)}$$

($\boldsymbol{\omega_i}$) – элементарное событие из множества элементарных событий Ω, из вероятностного пространства (Ω, σ, P).

Глава 2. Описания используемых методов.

2.1. Симплекс-метод.

Для решения данной выше задачи ЗЛП часто используют метод, суть которого состоит в построении таблицы из имеющихся данных и последующего решения задачи с помощью метода Жордана-Гаусса. Сам способ нахождения решения называется симплекс-методом **[4]**.

Для использования симплекс-метода необходимо привести задачу ЗЛП к каноническому виду:

максимизировать $z(x) = \sum_{k=1}^{m} c_k x_k$ при условии, что

$\sum_{k=1}^{m} a_{ik} x_k + s_k = b_i$

$x_k > 0$, $i = 1 \ldots n$

Далее составляем симплекс таблицу и выбираем первоначальный опорный план. В качестве него берутся переменные, значения которых неотрицательны, в то время как значения остальных из таблицы равны 0.

Под опорным планом понимается любое решение ЗЛП, которое удовлетворяет системе ограничений.

	x_1	…	x_n	s_1	…	s_n	
s_1	a_{11}	…	a_{1m}	1	…	0	b_1
…	…	…	…	…	…	…	…
s_n	a_{n1}	…	a_{nm}	0	…	1	b_n
z	$-c_1$	…	$-c_k$	0	…	0	b_0

Затем начинаем руководствоваться алгоритмом симплекс метода:

1. Проверяем, чтобы хоть 1 элемент из c_k был отрицательным. Если все элементы положительны – решение найдено.
2. Выбираем ведущий столбец. Для этого среди столбцов с коэффициентами целевой функции (c_k), которые меньше 0, выбираем

тот, в котором этот элемент минимален. Этот столбец будет ведущим. Для выбора ведущей строки, находим b_k/a_{ik}, где k – номер ведущего столбца и выбираем ту строку, в которой частное будет минимальным, при условии, что $a_{ik} > 0$.

3. Меняем местами переменные ведущего столбца и ведущей строки.
4. С помощью метода Гаусса делаем ведущий элемент равным 1, а все остальные коэффициенты из ведущего столбца равными 0.

В случае, когда задача линейного программирования будет рандомизирована, симплекс-таблицы будут иметь следующий вид:

	x_1	…	x_n	s_1	…	s_n	
s_1	$a_{11}(\omega)$	…	$a_{1m}(\omega)$	1	…	0	$b_1(\omega)$
…	…	…	…	…	…	…	…
s_n	a_{n1}	…	$a_{nm}(\omega)$	0	…	1	$b_n(\omega)$
z	$-c_1(\omega)$	…	$-c_k(\omega)$	0	…	0	$b_0(\omega)$

2.2. Сведения из теории вероятности.

Пусть есть вероятностное пространство {Ω, σ, P}.

Случайная величина – измеримая функция $\xi = \xi(\omega)$, которая является отображением Ω→R.

Для того, чтобы описать случайную величину, нужно задать закон распределения, в качестве которого выступает соотношение, устанавливающее связь между областью возможных значений (реализациями) случайной величины ξ и соответствующими ей вероятностями P.

Закон распределения определяется функцией $F(x) = P(X < x)$, которая выражает вероятность события $X < x$, где x является текущей переменной.

Производная от функции распределения называется **плотностью распределения**. Она имеет следующий вид:

$$f(x) = F'(x)$$

Математическое ожидание непрерывной случайной величины – интеграл вида:

$$M(X) = \int_{-\infty}^{\infty} xf(x)dx$$

В частности, математическое ожидание характеризует среднее значение случайной величины.

Дисперсия непрерывной случайной величины:

$$D(X) = \int_{-\infty}^{\infty}[x - M(x)]^2 f(x)dx = M\{[x - M(x)]^2\}$$

Дисперсия характеризует собой разброс значений случайной величины от её математического ожидания.

Более наглядным показателем разброса является **среднее квадратическое отклонение**, которое является квадратным корнем из дисперсии:

$$\sigma(X) = \sqrt{D(X)}$$

Случайная величина распределена по нормальному закону распределения, если её плотность имеет вид:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}},$$

где $\sigma -$ среднее квадратическое отклонение , а – математическое ожидание.

Неравенство Чебышева: $P(|X - M[X]| < \varepsilon) \geq 1 - \frac{D[X]}{\varepsilon^2}$

Закон больших чисел: Пусть $\xi_1 \ldots \xi_n$ попарно независимые случайные величины, их математическое ожидание равно a, дисперсии ограничены равномерно, то $\lim_{n\to\infty} P\left(\left|\frac{\sum_{i=1}^{n} \xi_i}{n} - a\right| < \varepsilon\right) = 1$, при любом $\varepsilon > 0$.

В оценке (вероятностной) отклонений (1.7) используется **правило трёх сигм**, суть которого состоит в следующем: вероятность того, что некоторая случайная величина ξ с нормальным законом распределения отклонится от своего математического ожидания (M_ξ) более, чем на тройное среднее квадратическое отклонение (3σ), равна 0,003, т.е.

$$P(|\xi - a| < 3\sigma)=0{,}997$$

2.3. Оценки представительности выборки при использовании метода статистических испытаний.

Определим нужное количество испытаний *n* для розыгрыша случайной величины так, чтобы вероятность отклонений величины по вероятности отличалась на заданное значение **[5, 6]**.

Далее будет использоваться неравенство Чебышева:

$$P\{|\xi_n - M(\xi_n)| \geq \varepsilon\} \leq \frac{M\{|\xi_n - M(\xi_n)|^2\}}{\varepsilon^2} = \frac{D(\xi_n)}{\varepsilon^2}$$

которое является следствием из центральной предельной теоремы.

Из неравенства Чебышева видно, что если

$$\lim_{n\to\infty} D(\xi_n) = 0,$$

то среднее арифметическое рассматриваемых значений ξ сходится по вероятности к своему математическому ожиданию. Тогда мы получаем, что при фиксированном любом $\varepsilon > 0$

$$P\{|\xi_n - M(\xi_n)| \geq \varepsilon\} \to 0, \text{при } n \to \infty,$$

то есть обратное неравенство:

$$\{|\xi_n - M(\xi_n)| \leq \varepsilon\}$$

имеет место.

Приведённые выше действия позволяют сказать, что повышая количество испытаний мы будем повышать точность оценки статистических характеристикm (*M*, *D*).

2.4. Метод статистических испытаний и алгоритм корректировки плана.

Для решения поставленной задачи проводится серия из *N* испытаний, для того, чтобы дать оценки для математического ожидания и дисперсии **[7]**. В каждом испытании коэффициенты будут меняться. Получим *N* новых значений целевой функции. Как описывалось ранее в главе 1 новые параметры в ЗЛП имеют следующий вид:

$$a_{ik}(\omega) = a_{ik} \pm \Delta a_{ik}$$

$$c_k(\omega) = c_k \pm \Delta c_k$$

$$b_i(\omega) = b_i \pm \Delta b_i \ ,$$

где $\Delta a_{ik}, \Delta c_k, \Delta b_i$ есть случайные величины, распределённые по нормальному закону распределения на интервале

[-3σ,3σ]. Для удобства обозначим Δ=3σ, тогда новый промежуток получается [-Δ,Δ], где $\Delta = \frac{max_i b_i}{t}$, где *t* – некоторое целое число, которое можно выбрать любым. Понятно, что с увеличением *t* также будет уменьшаться область значений σ (Δ).

Сначала решается детерминированный случай с помощью симплекс метода. При этом получаем оптимальный план *x* $(x_1 \ldots x_i)$, и оптимальное значение целевой функции *z(x)*. Далее меняем коэффициенты в исходной задаче линейного программирования согласно (1.7) и также с помощью симплекс-метода получаем *n* значений целевой функции *z(x)*. Из *n* полученных значений *z* можем сделать статистические оценки математического ожидания и дисперсии:

$$M^*(z) = \frac{1}{N}\sum_{i=1}^{N} z_i \qquad (\mathbf{2.1})$$

$$D^*(z) = \frac{1}{N-1}\sum_{i=1}^{n}(z_i - M^*(z))^2 \qquad (\mathbf{2.2})$$

Для анализа получившихся результатов важным моментом являются оценки математического ожидания и дисперсии, найденные в соответствии с (2.1-2.2). Мы можем выделить три основных случая:

1. Оценка математического ожидания не отличается от исходного оптимального значения целевой функции, а дисперсия стремится к нулю.
2. Оценка математического ожидания отличается не сильно от исходного значения z(x), а дисперсия достаточно мала.
3. Оценка математического ожидания сильно отличается (меньше), чем исходное оптимальное значения, а дисперсия получается достаточно большой.

В случае, когда возникает 1 или 2 ситуация, понятно, что изначальный план также остаётся оптимальным. При выполнении 3 ситуации мы можем сделать вывод, что в случае таких отклонений требуется выбрать другой план в качестве оптимального для получения наибольших значений целевой функции.

В приложении 1 листинг написанной программы, которая позволяет проводить анализ, задавая коэффициенты *t* и *N*. Также программа выводит оценки математического ожидания и дисперсии, на основе которых можно сделать выводы об оптимальности текущего плана. Также выводятся результаты при подсчёте всех симплекс-таблиц, соответственно имеется возможность выбрать другой план в качестве оптимального для максимизации прибыли.

Глава 3. Пример решения с помощью алгоритма.

Пусть некоторая компания занимается производством изделий из металла. Выпускается $n = 4$ видов изделий. Для изготовления каждого вида изделий требуются: металл на изготовление(кг), время на изготовление одного изделия(ч), клёпки(шт), электроды для сварки(шт). Для каждого вида изделия имеются ограничения по перечисленным требованиям. Все изделия имеют свою цену продажи за штуку. Цель задачи определить, сколько единиц каждого вида продукции должна изготовить компания, чтобы потратить все закупленные ресурсы и получить наибольшую прибыль. Сначала детерминированный случай решаем с помощью симплекс-метода. Для этого запишем используемую в примере ЗЛП, приведённую к каноническому виду:

Тип сырья / Вид продукта	**Металл, кг**	**Клёпки, шт**	**Электроды, шт**	**Время, ч**	**Цена, тыс. руб.**
1	25	20	4	28	30
2	23	6	6	32	45
3	14	0	7	17	12
4	70	4	10	22	65
ограничения	2500	1500	1100	1280	

Далее внесём данные в симплекс-таблицу:

	x_1	x_2	x_3	x_4	s_1	s_2	s_3	s_4	
s_1	25	23	14	70	1	0	0	0	2500
s_2	20	6	0	4	0	1	0	0	1500
s_3	4	6	7	10	0	0	1	0	1100
s_4	28	32	17	22	0	0	0	1	1280
z	-30	-45	-12	-65	0	0	0	0	

решим её и получим $z(x) = 2793$ – наибольшее значение целевой функции в детерминированном случае, а оптимальный опорный план будет следующим: $x(0, 20, 0, 29)$.

После этого проведём серию из 10000 испытаний, в каждом из которых каждый раз будут меняться коэффициенты a_{ik}, b_i, c_k, то есть добавляем элемент случайности. В качестве коэффициента t выберем 10.

Получим оценки математического ожидания $M^*(z) = 2712$ и дисперсии $D^*(z) = 8656$. Полученные оценки говорят о том, что изначальный план не будет оптимальным при заданных отклонениях. Оценивая результаты, стоит обратить внимание на событие, где мы получили $z^* = 2516$ с новым оптимальным планом $x = (0, 16, 0, 28)$.

Далее повторим вышеописанные действия, но возьмём уже $t = 50$. Соответственно получим новые оценки для $M^*(z)$ и $D^*(z)$, которые будут равны соответственно 2785 и 479. Полученные в этом случае результаты говорят о том, что оптимальный план $x(0, 20, 0, 29)$ детерминированной ЗЛП будет оптимальным также и далее, т.к. мы получили достаточно близкое значение оценки математического ожидания к значению целевой функции $z(x)$ и значение $D^*(z)$ также сравнительно небольшое.

Заключение

В данной работе рассмотрена корректировка предварительного плана в задачах линейного программирования в условиях, когда значения стоимостных параметров заранее неизвестны. Произведена оценка оптимальности изначального плана для последующих, заранее неизвестных, ЗЛП с коэффициентами, которые меняются по нормальному распределению.

Список литературы.

1. Красс М. С., Чупрынов Б.П. Математика для экономистов. СПБ.: Питер, 2005. 464 с.
2. Красс М. С., Чупрынов Б.П. Математические методы и модели для магистрантов экономики. СПБ.: Питер, 2010. 496 с.
3. Колбин В.В. Стохастическое программирование. Palmarium Academic Publishing, 2013. 396 с.
4. Зуховицкий С. И., Авдеева Л.И. Линейное и выпуклое программирование. М.: Наука, 1967. 460 с.
5. Гмурман В. Е. Теория вероятностей и математическая статистика. М.: Высшая школа, 1972. 368 с.
6. Вентцель Е. С. Теория вероятностей. М.: Наука, 1969. 576 с.
7. Ермаков С. М. Метод Монте-Карло и смежные вопросы. М.: ФИЗМАТЛИТ, 1975. 472 с.

Приложение 1. Листинг программы.

```
import dipl as d

import random as r

p = 0

time = 0

disp = 0

mat = 0

s = 0


#Основная часть

a = [[2500, 25, 23, 14, 70, 1, 0, 0, 0],

    [1500, 20, 6, 0, 4, 0, 1, 0, 0],

    [1100, 4, 6, 7, 10, 0, 0, 1, 0],

    [1280, 28, 32, 17, 22, 0, 0, 0, 1],

    [0, -30, -45, -12, -65, 0, 0, 0, 0]]


print(' ORIG:')

for i in a:

   print(i)


b = d.optimal(a)

x1,x2,x3,x4,zz = d.answer(b)

print('SOLVED ORIG:')

for i in b:

   print(i)
```

```
print('x1=',x1,'x2=',x2,'x3=',x3,'x4=',x4,'z=',zz)
k = int(input('ВВедите коэфф. t = '))
while k == 0:
    k = int(input('введите t > 0: '))
n = int(input('ВВедите количество испытаний:'))
z1 = []
for i in range(n):
    z1.append(0)
for i in range(n):
    a = [[2500, 25, 23, 14, 70, 1, 0, 0, 0],
         [1500, 20, 6, 0, 4, 0, 1, 0, 0],
         [1100, 4, 6, 7, 10, 0, 0, 1, 0],
         [1280, 28, 32, 17, 22, 0, 0, 0, 1],
         [0, -30, -45, -12, -65, 0, 0, 0, 0]]
    e = a
    e,f,f1 = d.randomik(x1,x2,x3,x4,a,k)
    zz = -round(e[4][1])*x1 -round(e[4][2])*x2 -round(e[4][3])*x3 -round(e[4][4])*x4
    b = d.optimal(e)
    xx1,xx2,xx3,xx4,z = d.answer(b)
    print (xx1,xx2,xx3,xx4, 'z = ',z)
    if (xx1 == 0 and xx2 == 0 and xx3 == 0 and xx4 == 0) or z == 0:
        s = 0
        z1[i] = 0
        print('net')
    else:
```

```
        if z == zz:
            time += 1
        s = s + z
        z1[i] = z
mat = s/n
for i in range(n):
    p = p +(z1[i] - mat)**2
disp = p/(n-1)
print('M(z)',mat, 'D(z)',disp)


#Файл с функциями
import random as r
def cheq(f,m):
    c = 0
    for i in m[-1]:
        if i < 0:
            c +=1;
    if c == 0:
        f = False
    return f
```

```
def poisk(m): # ищем номер и значение элемента
    s = m[-1][0]
    g =0.001
    f = 100000
    for i in m[-1]:
        if i<0 and i < s:
            s = i
            c = m[-1].index(s)
    for i in m:
        if i[c] != 0:
            if (i[0])/(i[c])>0:
                if f > (i[0])/(i[c]):
                    f = (i[0])/(i[c])
                    g = i[c]
                    cn = m.index(i)
    return cn,c,g


def key(cn,c,s,m):#Ключ, то есть меняем строку, чтоб элемент стал 1
    for j in range(len(m[cn])):
        m[cn][j] = m[cn][j]/s
    s = m[cn][c]
    return s,m


def hod(cn,c,m,s): #один шаг
    for i in range(len(m)):
```

```
            if i!=cn:
                s1=m[i][c]
                for j in range(len(m[i])):
                    m[i][j] = m[i][j] - (m[cn][j])*s1
    return m


def optimal(m):#основная функция
    flag = True
    o = 0
    for i in m[-1]:
        if i < 0:
            o+=1
    if o == 0:
        flag = False
    while flag == True:
        cn,c,s = poisk(m)
        s,m = key(cn,c,s,m)
        m = hod(cn,c,m,s)
        flag = cheq(flag,m)
    for i in range(len(m)):
        for j in range(len(m[i])):
            o = m[i][j]
            m[i][j] = round(o)
    return m
```

```
def fx1(cn,m):
    k = 0
    x1 = 0
    if cn == 1:
        for i in range(len(m)):
            if m[i][cn] == 0:
                k+=1
            elif m[i][cn] == 1:
                s = cn
                t = i
        if k == 4:
            if s == 1:
                x1 = m[t][0]
            else:
                x1 = 0
    return x1


def fx2(cn,m):
    k = 0
    x2 = 0
    if cn == 2:
        for i in range(len(m)):
            if m[i][cn] == 0:
                k+=1
```

```
            elif m[i][cn] == 1:
                s = cn
                t = i
        if k == 4:
            if s == 2:
                x2 = m[t][0]
            else:
                x2 = 0
    return x2


def fx3(cn,m):
    k = 0
    x3 = 0
    if cn == 3:
        for i in range(len(m)):
            if m[i][cn] == 0:
                k+=1
            elif m[i][cn] == 1:
                s = cn
                t = i
        if k == 4:
            if s == 3:
                x3 = m[t][0]
            else:
                x3 = 0
```

```
    return x3


def fx4(cn,m):
    k = 0
    x4 = 0
    if cn == 4:
        for i in range(len(m)):
            if m[i][cn] == 0:
                k+=1
            elif m[i][cn] == 1:
                s = cn
                t = i
        if k == 4:
            if s == 4:
                x4 = m[t][0]
            else:
                x4 = 0
    return x4


def answer(m):
    z = m[-1][0]
    x1 = 0
    x2 = 0
    x3 = 0
    x4 = 0
```

```
    for i in range(len(m)):
        for j in range(len(m[i])):
            if m[i][j] == 0:
                s = m[i][j]
                cn = j
                if cn == 1:
                    x1 = fx1(j,m)
                elif cn == 2:
                    x2 = fx2(j,m)
                elif cn == 3:
                    x3 = fx3(j,m)
                elif cn == 4:
                    x4 = fx4(j,m)
    return x1,x2,x3,x4,z



def randomik(x1,x2,x3,x4,m,k):
    b = [0,0,0,0]
    x = [x1,x2,x3,x4]
    f = 0
    pr = 1000000
    for i in range(len(m)):
        for j in range(len(m[i])):
            if j == 0:
                if i <= 3:
```

```
            if pr >= m[i][j]/k:
                pr = m[i][j]/k
for i in range(len(m)):
    for j in range(len(m[i])):
        if j == 0:
            if i <= 3:
                b[i] = r.normalvariate(0, (pr/3))
                o = m[i][j] + b[i]
                m[i][j] = o
            if i == 4 and j == 0:
                m[4][0] = 0
for i in range(len(m)):
    for j in range(len(m[i])):
        if j > 0 and j <= 4:
            if i <= 3:
                if x[j-1] != 0:
                    f = (b[i]**2)/(x[j-1]**2)
                    o = m[i][j] + f**0.5
                    m[i][j] = o
                else:
                    o = m[i][j]
                    m[i][j] = o
for i in range(len(m)):
    for j in range(len(m[i])):
        if j > 0 and j <= 4:
```

```
            if i == 4:
                o = m[i][j] + r.normalvariate(0, (((m[i][j])/k)/3))
                m[i][j] = o
    for i in range(len(m)):
        for j in range(len(m[i])):
            o = m[i][j]
            m[i][j] = round(o)
    return m,x,b


def random(m):
    for i in range(len(m)):
        for j in range(len(m[i])):
            if j<=2:
                if i == 3:
                    o = (m[i][j]) - r.random()
                    m[i][j] = o
                o = (m[i][j]) + r.random()
                m[i][j] = o
                if i == 3 and j == 0:
                    m[3][0] = 0
    return m
```